Software Engineering Learning in HFOSS: A Multi-Institutional Study

Heidi J. C. Ellis, Western New England University

Heidi Ellis is Chair and Professor in the Computer Science and Information Technology department at Western New England University. Dr. Ellis has a long-time interest in software engineering education and has been interested in student participation in Humanitarian Free and Open Source Software (HFOSS) since 2006.

Dr. Gregory W Hislop, Drexel University (Eng. & Eng. Tech.)

Gregory Hislop is a Professor and Senior Associate Dean in the College of Computing and Informatics at Drexel University. His scholarly interests span computing education research, information technology for teaching and learning, and software engineering. Prior to coming to Drexel, Dr. Hislop spent eighteen years working in government and industry where his efforts included software development and support, technology planning and evaluation, and development and delivery of technical education.

Dr. Sarah Monisha Pulimood, The College of New Jersey

S. Monisha Pulimood is on the faculty of the Department of Computer Science at The College of New Jersey. She has been successfully incorporating immersive learning experiences and multidisciplinary collaborative projects into her courses for several years; has published on undergraduate research, collaboration, project management, and diversity; is PI of ”Collaborating Across Boundaries to Engage Undergraduates in Computational Thinking” (NSF-DUE Award #1141170) and co-PI of ”Scholarships for Success in Computational Science” (NSF-DUE Award #1356235, PI Thomas Hagedorn).

Dr. Becka Morgan, Western Oregon University

Becka Morgan takes great joy in teaching students computing languages, a subject she has been passionate about since she learned to program in 2006 as a non-traditional student. She is driven to create an inclusive environment. Consequently Dr. Morgan was drawn to teaching FOSS and HFOSS development based on work that is being done that suggests underrepresented groups are attracted to HFOSS participation. She teaches a one term HFOSS course to both senior and graduate level students. The goal of the course is to engage all students in participation that ranges from improving documentation to submitting patches. Learning to teach students how to participate in HFOSS is an ongoing process. As part of the continuing efforts to pursue that knowledge Becka is a graduate of the 2013 POSSE workshop.

Dr. Ben Coleman, Moravian College

Ben Coleman is a member of the Department of Mathematics and Computer Science at Moravian College, a small, liberal arts college in eastern Pennsylvania. His research interests are in software engineering, particularly in areas related to pedagogy and bringing real-world, hands-on experience to undergraduates.

Software Engineering Learning in HFOSS: A Multi-Institutional Study

Heidi J. C. Ellis
Dept. of CS & IT
Western New England University
Springfield, MA 01119
ellis@wne.edu

Gregory W. Hislop
College of Computing and Informatics
Drexel University
Philadelphia, PA
hislop@drexel.edu

S. Monisha Pulimood
Dept. of Computer Science
The College of New Jersey
Ewing, NJ 08628
pulimood@tcnj.edu

Becka Morgan
Dept. of Computer Science
Western Oregon University
Monmouth, OR 97361
morganb@wou.edu

Ben Coleman
Dept. of Mathematics and Computer Science
Moravian College
Bethlehem, PA 18018
coleman@cs.moravian.edu

Abstract

Real-world projects are frequently used to provide students with professional software development experience. Involvement in Humanitarian Free and Open Source Software (HFOSS) projects allows students to learn about a complex software project within a community of professionals. In addition, the humanitarian aspect of HFOSS provides students with the motivation of developing software that will “do good”. The opportunities for learning in such an environment range from technical topics to communication to professionalism and more. This paper reports on the results of a multi-institution study of student perceptions of learning within an HFOSS project. The study expands an earlier study[1] and involves four different institutions with courses offered between fall 2013 and fall 2014. Students were involved in projects including GNOME MouseTrap, a project to provide alternative input for users with disabilities, and OpenMRS, an electronic medical record system used extensively in developing countries. Results generally support the outcomes of the early study, but provide stronger evidence that student involvement in HFOSS promotes student learning in the areas of tools and techniques and technical knowledge about the process and tools used to develop an HFOSS project.

1. Introduction

Software engineering programs as well as most computer science programs desire to provide students with experience working on a real-world project in order to supply students with an understanding of professional practice including such skills as teamwork, communication, work ethic, self-confidence and more. In fact, the SE 2004 curriculum guidelines[2] emphasize the need for including professional practice in the education of software engineers. A common way to provide this experience is through instructor and/or student defined projects[3-7]. However this approach lacks a professional community from which students can learn. Alternatively, faculty can engage an industry partner to provide professional interaction[8-10]. While this approach has the advantage of providing students with industry experience, the disadvantages include that most of the development artifacts are protected and students typically cannot use their project as evidence of their accomplishments to potential employers.

Student involvement in Free and Open Source Software (FOSS) provides the opportunity to participate in the development of a sizeable real-world project while interacting with a professional community. FOSS projects are characterized by community development, openness to contributions, transparency and accessibility of artifacts, distributed and global development, and meritocratic decision-making. As a result, students have the opportunity to observe and participate in many aspects of software development, which provides excellent educational opportunities[11-13]. Through FOSS projects, students can learn new tools and programming languages, gain professional experience, and create a professional network[14]. Specifically, students can engage the full range of software engineering activities through documentation, design, coding, testing, maintenance, etc.[15]

Humanitarian FOSS (HFOSS) is FOSS that somehow improves the human condition, with applications ranging from healthcare to education to disaster management and more. Similar to FOSS projects, involvement in HFOSS allows students to advance a range of technical skills[16] and improve professional skills.[17] HFOSS also provides students with a variety of learning approaches including active learning, problem-based learning, and collaborative learning[18]. Other benefits include exposure to a project of significant complexity, increased awareness of the benefits that can be provided by computing, and the advantage of having open source experience on a student's resume. HFOSS holds the added attraction of allowing students to "do good" which studies have shown is motivating to students[1, 19, 20].

However, these benefits come with some cost. The learning environment within an HFOSS project is less structured than the typical classroom and some students may have difficulties adjusting to this more flexible learning approach. HFOSS also presents a series of learning curves for both faculty members and students including tools, approaches, domain knowledge, FOSS culture, and professional interactions. Faculty members may face challenges as they negotiate communication and support with the HFOSS community to select an appropriate project, identify student contributions, and fit course schedules with project release schedules. FERPA, intellectual policy rules, and institutional requirements may place additional constraints on student participation in HFOSS projects.

Despite these challenges, there are multiple instances of successful student involvement in HFOSS. Liu[21] is one of the earliest published reports of an effort to involve students in HFOSS projects via service learning. Ding[22] discusses a virtual service learning model that involves students in FOSS projects to aid in learning of professional communication and documentation, and Jacobs[23] presents an approach to involve students in the development of games for the One Laptop Per Child project. Finally, a multi-institutional effort to developing real-world HFOSS projects is presented by MacKellar, Sabin, and Tucker[24].

Student involvement in HFOSS projects has been tracked since 2006[25]. Initial studies indicate that benefits from involvement in HFOSS projects include greater student motivation to pursue computing careers and an increase in software engineering knowledge[1, 26]. In fact, involvement in HFOSS is increasingly being utilized as a way to educate software engineering students and there are a growing number of faculty members who are involving students in HFOSS projects (foss2serve.org).

This paper expands on an earlier effort[1] to report on a multi-institutional study on the impact of student participation in HFOSS.

2. The Institutions

Table 1 below outlines the four institutions involved in the study. These institutions are a mix of public and private institutions and are small to medium in size. The remainder of this section provides a brief overview of the institutions involved in the study to provide context for understanding the study.

Institution ID	Institution Size	Department	Department Size
A	1,500 undergrads	Math & CS	25 CS and 40 Math majors
B	6,100 undergrads, 800 grad students	CS & IS	245 CS and IS majors 58 MIS graduate students
C	2,500 undergrads, 1,000 grad students	CS & IT	100 CS and IT majors
D	6,100 undergrads	CS	125 CS majors

Table 1. Participating Institutions

Institution A is a small, private liberal arts college with approximately 1,500 undergraduate students. The department of mathematics and computer science has approximately 25 CS majors, and the CS major is designed to offer students hands-on experience on real-world projects while providing a grounding in theoretical ideas. The course used in the survey is a senior-level software engineering capstone course that used OpenMRS as a project base. Students addressed bug reports to learn the system and then designed and implemented a new add-on module.

Institution B is a public liberal arts university serving approximately 6,100 undergraduate students and 800 graduate students. Founded in 1980, the Computer Science program is software-oriented and follows the ACM Curriculum Guidelines. The Computer Science program is also closely aligned with the current needs of industry. The course used in this study is a senior level software engineering elective course that applies toward the software engineering track within the CS major. The course uses an existing FOSS project with the goal to engage students in a real world project that will provide job skills that pertain to the “real” world. Although the focus was on the entirety of the project, including documentation, bug triage, activism, and translation, students focused on fixing bugs, testing, or coding to fulfill the software engineering elective requirement.

Institution C is a small, private institution with approximately 2,500 undergraduates. The Computer Science and Information Technology department has around 100 students and offers BS degrees in Computer Science and in Information Technology. Both degree programs are fashioned after the ACM Curriculum Guidelines for those degrees. The course used for the study is a Software Engineering course that is taken by all computer science majors in the fall of their senior year. The course is organized around an ongoing HFOSS project and is intended to expose students to the major software development activities including requirements, design, implementation, test and maintenance.

Institution D is a public, residential, primarily undergraduate institution with approximately 6,100 students. The Department of Computer Science currently has about 125 students and provides a comprehensive learning environment through a rigorous curriculum designed to meet the needs of students interested in both careers in the industry and graduate school. The main course used in this study was Software Engineering, which is required for all majors and is typically taken in the junior or senior year. In this course, students collaborate on a large project in teams, where they apply concepts learned. In recent semesters, students have been working on an HFOSS, web-based system that manages data on brownfields and legislation related to pollution and the environment, for Habitat for Humanity and citizens of the area.

3. The Study

Student participation in HFOSS has been studied since 2006[25]. However, the investigation of student opinion of involvement in HFOSS started with a handful of small, liberal arts institutions. In recent years, student involvement in HFOSS has expanded to a larger number of colleges and universities as institutions understand the benefits of students learning within the environment of a real-world project that improves the human condition[27]. The study discussed in this paper expands an existing study into the impact of student participation in HFOSS projects[1, 28] and includes three institutions which have recently incorporated student involvement in HFOSS, in addition to one institution from the previous study. The larger study investigates three aspects of the impact of student participation in HFOSS:

1. The impact of participation in an HFOSS project on student attitude towards computing.
2. The degree of perceived learning related to software engineering knowledge and skills.
3. The impact of participation in an HFOSS project on major selection and career plans.

This paper reports on the results across four different institutions for the second aspect on software engineering. Table 2 below summarizes the courses that were involved in the study.

Course	**Term Offered**	**Number of Students**	**Length of Term**
Senior Capstone	Spring 2014	10	15 weeks
Open Source Software Development	Winter 2014	20	10 weeks
Software Engineering	Fall 2013, Fall 2014	6 8	15 weeks
Software Engineering	Fall 2013	19	15 weeks

Table 2. Courses in the Study

The courses used in the study were relatively similar in being upper-level, team project courses focused on software engineering topics. One institution has 10 week quarter terms and the others have 15 week semesters.

The study presented in this article focuses on whether participation in HFOSS projects impacts student perception of software engineering learning. The hypotheses for the study are:

> **H_o**: Student involvement in an HFOSS project has no impact on perceived learning of software engineering knowledge
> **H_a**: Student involvement in an HFOSS project has a positive impact on perceived learning of software engineering knowledge

The study instrument includes background information such as student age and experience as well as a five-point Likert scale survey with response values “strongly disagree”, “disagree”, “neutral”, “agree”, and “strongly agree.” “Don’t know” and “Not applicable” options were also included. The survey contains three sections of Likert items, one section for each of the three aspects under study. Note that Likert scale items allow for both agreement and disagreement. Table 3 below contains sample survey items for the three aspects under study. The “H6” item relates to student motivation (aspect 1), the “SE2” item relates to perceived software engineering learning (aspect 2) and the “G2” item relates to impact on major and career plans (aspect 3). The survey items are worded so that the positive outcome, H_a, will be reflected by student agreement (“agree” or “strongly agree”) with each statement.

ID	**Item**
H6	Working with an H-FOSS community to develop a project has increased my interest in computing.
SE2	I am comfortable that I could participate in the planning and development of a real-world software project.
G2	Participation in an H-FOSS project has positively reinforced my decision to make computing my major.

Table 3. Example Survey Items

This paper focuses on the software engineering aspects of the results. Table 4 below contains the software engineering related survey items.

ID	**Item**
SE1	I am comfortable that I could participate in the planning and development of a real-world software project.
SE2	I can list the steps in the software process we used in HFOSS project.
SE3	I can use a software process to develop an HFOSS project.
SE4	I am sure that I can actively participate in an HFOSS community to develop a software project.
SE5	I have gained some confidence in collaborating with professionals from a variety of locations and cultures.
SE6	I can describe the impact of project complexity on the approaches used to develop software.
SE7	I can describe the impact of project size on the approaches used to develop software.
SE8	I am confident that I can maintain an HFOSS project.
SE9	I can describe the drawbacks and benefits of FOSS to society.
SE10	I can use all tools and techniques employed in my HFOSS project.
SE11	I can participate in an HFOSS development team’s interactions.
SE12	Participation in an HFOSS project has improved my understanding of how to behave like a computing professional.

Table 4. Software Engineering Post-Course Survey Items

There are some differences in how the surveys were administered as one site only included questions SE4, SE5, and SE8-SE12 and another site had a fewer number of post-course responses than pre-course responses. The Likert items in the surveys were converted to an ordinal number from one to five with one representing the “strongly disagree” response, five representing the “strongly agree” response, and three representing the “neutral” response.

4. Results and Discussion

This section presents and discusses results of the survey. The section begins with a summary of basic demographic data for the student population. This is followed by an analysis of pre- and post-course results, as well some consideration of gender and the impact of prior programming ability.

Student Population

The students in the study samples are fairly typical of the population of computing majors in U.S. degree programs. These programs serve primarily traditional-age undergraduate students, with 86% in the sample being aged 18-23 and 14% older than 23. 81% of the students are male and 75% are white. While a variety of majors are represented in the classes, 80% are computer science majors, and 96% are majoring in some computing discipline (CE, CS, IS, IT, or MIS).

Pre- and Post-Course Software Engineering Capability

Likert scale items SE1 – SE12, shown in Table 4, gather student self-perception about their software engineering knowledge and skill. The data show significant shift in student response when comparing the pre- and post-course surveys. To provide an overall summary of this shift, Figure 1 presents the pre- and post-course responses. Because the number of responses on each survey is different, the data is presented using percentages. The height of each bar represents the percent of students who either agreed or strongly agreed with each item. For example, for SE2, only 29% of students agreed pre-course that they could list the steps in the software process used to develop an HFOSS project, and 71% agreed post-course.

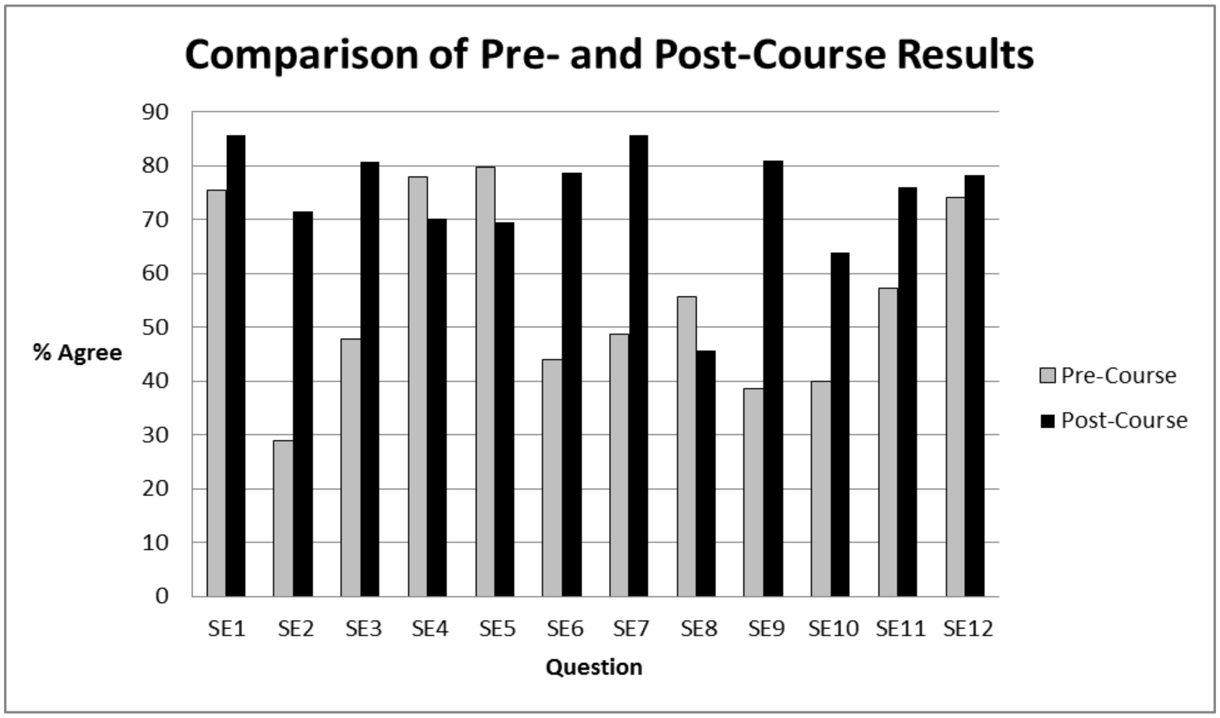

Figure 1. Percentage of Students Agreeing with Each SE Survey Item Pre- and Post-course

Figure 1 shows some very substantial changes in the percent of students agreeing with many of the 12 SE survey items. In general these large changes are associated with the items that had the lowest percent agreement pre-course. Overall, these shifts provide strong initial evidence of positive outcome from the HFOSS experience.

Table 5 below provides a different perspective on the pre- and post-course results. The column labelled "Net %" shows the difference between the pre- and post-course agreement responses for each item. That is, the difference in the height of each pair of bars in Figure 1. For example, for SE2, the 29% agreement pre-course, and 71% agreement post-course result in the 42 in Table 5 (71% - 29%).

ID	**Item**	**Net %**	***p*-Value**
SE1	I am comfortable that I could participate in the planning and development of a real-world software project.	10	0.24
SE2	I can list the steps in the software process we used in HFOSS project.	42	< 0.01 **
SE3	I can use a software process to develop an HFOSS project.	33	< 0.01 **
SE4	I am sure that I can actively participate in an HFOSS community to develop a software project.	-8	0.29
SE5	I have gained some confidence in collaborating with professionals from a variety of locations and cultures.	-10	0.05 **
SE6	I can describe the impact of project complexity on the approaches used to develop software.	35	0.01 **
SE7	I can describe the impact of project size on the approaches used to develop software.	37	0.01 **
SE8	I am confident that I can maintain an HFOSS project.	-10	0.13
SE9	I can describe the drawbacks and benefits of FOSS to society.	42	<0.01 **
SE10	I can use all tools and techniques employed in my HFOSS project.	24	0.01 **
SE11	I can participate in an HFOSS development team's interactions.	19	0.02 **
SE12	Participation in an HFOSS project has improved my understanding of how to behave like a computing professional.	4	0.39

Table 5. Software Engineering Survey Items, Pre- and Post-Course Comparison

The "*p*-Value" column of Table 5 provides a significance measure of the difference between the pre- and post-course surveys. Since there is always a question as to the validity of treating Likert scale items as interval or ratio data, this analysis treats the data as ordinal and uses the non-parametric Mann-Whitney U to compare the two survey results and compute the significance measure. Items where the difference between pre- and post-course surveys is significant at the 0.05 level or better are marked with "**" for convenience. As can be seen, 8 of the 12 items show a significant difference. Note that the Mann-Whitney U was based on the full set of responses, ordered as 1, "strongly disagree," though 5, "strongly agree," and not on the percentage data shown in Figure 1 or the "Net %" column of Table 5.

Figure 1 and Table 5 in combination present a strong case that there is a significant positive outcome from the HFOSS courses, supporting research hypothesis H_a: Student involvement in an HFOSS project has a positive impact on perceived learning of software engineering knowledge. In addition, these results present stronger evidence for the results found in the initial work[1]. There are interesting aspects in the details of these data. First, in looking at the survey items that show significant results, some of the largest net % changes are found in items that tend to represent more specific skills or lower levels in the Bloom taxonomy. For example, SE2, ("I can list"), and SE6, SE7, and SE9, ("I can describe"), all fit that observation.

However, large net changes for some statements also offer evidence related to more advanced abilities. For example, the response to SE3, *I can use a software process to develop an HFOSS project*, SE10, *I can use all tools and techniques employed in my HFOSS project*, and SE 11, *I can participate in an HFOSS development team's interactions*, in combination provide strong evidence that students have confidence in their ability to plan and develop a real-world software project after participating in an HFOSS project.

Second, it is important to note that the pre- to post-course change was not in the direction of increased agreement for all survey items. Recall that the items were all worded so that agreement was the desirable post-course outcome. Three items, SE4, SE5, and SE8, show lower agreement post-course than pre-course. For two of those items, the change is not significant at the 0.05 level, and the third is right at the 0.05 significance level. Even so, it seems worth considering what these items may have in common. While only one had significance in showing lower student agreement, all three represent cases of no significant change in the positive direction – that is, toward more student agreement with the item post-course. In this sense, these items are like SE1 and SE12, which showed positive change, but not significant at 0.05.

Looking at these 5 items, SE1, SE4, SE5, SE8, and SE12, in Figure 1 is instructive. It turns out that four of the five (SE1, SE4, SE5, and SE12) were the four survey items with the highest agreement pre-course, with all four registering over 70% agreement pre-course. Given these high pre-course values, it seems unsurprising that the post-course value would not show significant change. This may also account for the drop in agreement with items SE4 and SE5 post-course. It seems possible that the students were actually over-confident of their ability pre-course, and that the HFOSS experience gave them a more realistic perspective on the collaboration and participation requirements of a large, distributed project.

Of the 5 items, SE8 fits the ideas above least well. The same effect of relatively high confidence pre-course could be part of the explanation here too, given that SE8 has the sixth highest pre-course agreement. But the SE8 pre-course value (56%) is considerable below the 70+% of the other four. Also, SE8 is the one item where the drop in agreement is significant at the 0.05 level. It may be that the work involved with maintaining an HFOSS project simply seemed more daunting to students after exposure to an HFOSS project than before. It may also be that students have a more realistic view of what is required to participate in and maintain an HFOSS application at the end of the course than at the beginning. Such maintenance is a major responsibility, and the wording of the statement places that burden on the individual, compared to some of the other statements that address broad responsibilities, but are phrased in terms of participating or collaborating and imply a team effort more clearly.

Finally, the significant results are consistent with an earlier, smaller study that indicated significant increase in agreement for items SE2, SE3, SE6, SE7, and SE10[1]. The present study, with a wider range of institutions, and larger sample sizes adds to the set of survey items showing significantly more post-course agreement than in the initial study.

Gender Comparison

An investigation into the impact of gender on the results found one significant difference using a Mann-Whitney U: item 11 *I can participate in an HFOSS development team's interactions (p =*

0.001). This supports prior results[1] that indicate that females had a stronger response to items related to ability to participate in the planning and development of a real-world software project.

Programming Ability

An investigation of differences based on programming ability resulted in some interesting findings. Students with “low” programming skills were considered to have self-assessed a programming ability between one and three (67%). Students with “high” programming skills were considered to have self-assessed a programming ability of either four or five (32%). Two items showed a significantly stronger response for the high programming ability students: item SE4 *I am sure that I can actively participate in an HFOSS community to develop a software project (p = 0.02)* and item SE8 *I am confident that I can maintain an HFOSS project (p = 0.06)*. These results represent a subset of the results found in the initial study which indicated that students with a “high” programming ability showed significantly more agreement to a greater number of survey items. It should be noted that the percentage of students who self-assessed a “low” programming ability in the initial study was much lower (53%) when compared to the current study.

Perhaps the more interesting result is that there was **no** significant difference in student perception of ability to collaborate, ability to identify drawbacks and benefits of HFOSS, use of tools and techniques and understanding of professional interactions. This provides some evidence that success in software development is not dependent on programming ability alone.

5. Conclusion

The results of the study presented in this paper support research hypothesis **H_a: Student involvement in an HFOSS project has a positive impact on perceived learning of software engineering knowledge.** Results indicate that students feel that they gain considerable software engineering knowledge from participation in an HFOSS project ranging from tools and techniques to software process to understanding the impact of project size and complexity. Interestingly, some of the knowledge gained is related to learning on the higher levels of Bloom’s taxonomy. Results indicate a perceived increase in ability to use a software process to develop an HFOSS project and an increase in ability to participate in an HFOSS development team’s interactions. These results suggest that student participation in HFOSS has the potential to provide students with a holistic learning experience where learning ranges from factual recall to creation and evaluation.

Future work includes analysis of the humanitarian and career aspects of the data as well as investigation into direct measures of student learning via participation in HFOSS projects.

Acknowledgement

This material is based on work supported by the National Science Foundation under Grant Nos. - DUE-1225708, DUE-1225738, and DUE-1225688. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF).

Bibliography

1. Heidi J. C. Ellis, Gregory W. Hislop, Josephine Rodriguez, and Ralph A. Morelli. 2012. Student Software Engineering Learning via Participation in Humanitarian FOSS Projects. In Proceedings of the 119th Annual ASEE Conference and Exhibition, San Antonio, TX.
2. Software Engineering 2004 – Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. IEEE-CS/ACM. http://sites.computer.org/ccse/SE2004Volume.pdf 2004.Accessed 1/4/12.
3. Catherine Stringfellow and Divya Mule. 2013. Smartphone applications as software engineering projects. J. Comput. Sci. Coll. 28, 4 (April 2013), 27-34.
4. Sarah Monisha Pulimood and Ursula Wolz. 2008. Problem Solving in Community: A Necessary Shift in CS Pedagogy. In Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education (Portland, OR, USA, March 12 – 15, 2008). SIGCSE '08. ACM, New York, NY, 210-214.
5. Ursula Wolz and Sarah Monisha Pulimood. 2007. An Integrated Approach to Project Management through Classic CS III and Video Game Development. In Proceedings of the 38th Technical Symposium on Computer Science Education (Covington, Kentucky, March 7 – 10, 2007). SIGCSE '07. ACM, New York, NY, 322-326.
6. Tom Rishel. 2012. An innovative project structure for teaching software engineering. J. Comput. Sci. Coll. 28, 2 (December 2012), 232-237.
7. Samuel Mann and Lesley Smith. 2007. Software engineering class eating its own tail." Proceedings of the ninth Australasian Conference on Computing Education-Volume 66. Australian Computer Society, 115-123.
8. Adrian Rusu, Amalia Rusu, Rebecca Docimo, Confesor Santiago, and Mike Paglione. 2009. Academia-academia-industry collaborations on software engineering projects using local-remote teams. SIGCSE Bull. 41, 1 (March 2009), 301-305. DOI=10.1145/1539024.1508975 http://doi.acm.org/10.1145/1539024.1508975
9. Christopher K. Hobbs and Herbert H. Tsang. 2014. Industry in the Classroom: Equipping Students with Real-World Experience A reflection on the effects of industry partnered projects on computing education. In Proceedings of the Western Canadian Conference on Computing Education (WCCCE '14). ACM, New York, NY, USA, Article 7, 5 pages. DOI=10.1145/2597959.2597967 http://doi.acm.org/10.1145/2597959.2597967
10. Claudia Szabo. 2014. Student projects are not throwaways: teaching practical software maintenance in a software engineering course. In Proceedings of the 45th ACM technical symposium on Computer science education (SIGCSE '14). ACM, New York, NY, USA, 55-60. DOI=10.1145/2538862.2538965 http://doi.acm.org/10.1145/2538862.2538965
11. Robert Marmorstein. 2011. Open Source Contribution as an Effective Software Engineering Class Project. In Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education, ITiCSE '11, ACM, New York, NY, USA, 268-272.
12. Heidi J. C. Ellis, Gregory W. Hislop, Mel Chua, and Sebastian Dziallas. 2011. How to Involve Students in FOSS Projects, The 2011 Frontiers in Education Conference, IEEE. T1H-1,T1H-6, (Oct. 2011), 12-15, doi: 10.1109/FIE.2011.6142994
13. Stephanie Ludi. 2011. The benefits and challenges of using educational game projects in an undergraduate software engineering course, In Proceedings of the 1st International Workshop on Games and Software Engineering. ACM, New York, NY, USA. 13-16. 2011.
14. Eleni Stroulia, Ken Bauer, Michelle Craig, Karen Reid, and Greg Wilson. 2011. Teaching distributed software engineering with UCOSP: the undergraduate capstone open-source project. In Proceedings of the 2011 Community Building Workshop on Collaborative Teaching of Globally Distributed Software Development (CTGDSD '11). ACM, New York, NY, USA, 20-25. DOI=10.1145/1984665.1984670 http://doi.acm.org/10.1145/1984665.1984670
15. Clif Kussmaul, Heidi J. C. Ellis, and Gregory W. Hislop. 2012. 50 ways to be a FOSSer: simple ways to involve students & faculty (abstract only). In Proceedings of the 43rd ACM technical symposium on Computer Science Education (SIGCSE '12). ACM, New York, NY, USA, 671-671. DOI=10.1145/2157136.2157393 http://doi.acm.org/10.1145/2157136.2157393
16. Marisa Exter. 2014. Comparing educational experiences and on-the-job needs of educational software designers. In Proceedings of the 45th ACM technical symposium on Computer science education (SIGCSE '14). ACM, New York, NY, USA, 355-360. DOI=10.1145/2538862.2538970 http://doi.acm.org/10.1145/2538862.2538970
17. Rüdiger Glott, Andreas Meiszner, Sulayman K. Sowe, 2007. Report to FLOSSCom - Using the Principles of Informal Learning Environments of FLOSS Communities to Improve ICT Supported Formal Education: Phase 1 - Analysis of the Informal Learning Environment of FLOSS (Free/Libre Open Source Software)

Communities, http://www.academia.edu/2723574/FLOSSCom-Using_the_Principles_of_Informal_Learning_Environments_of_FLOSS_Communities_to_Improve_ICT_Supported_Formal_Education, Retrieved 11/29/2014.
18. Martin Weller, Andreas Meizsner, Sulayman K. Sowe, and Athanasis Karoulis. 2008. A Report to FLOSSCom - Using the Principles of Informal Learning Environments of FLOSS Communities to Improve ICT Supported Formal Education: Phase 2 - Report on the effectiveness of a FLOSS-like learning community in formal educational settings. http://flosshub.org/system/files/FLOSSCOM_Wp4_PHASE2_REPORT_d1.pdf Retrieved 11/29/14.
19. Rick Homkes. 2008. Assessing it service-learning. In Proceedings of the 9th ACM SIGITE conference on Information technology education (SIGITE '08). ACM, New York, NY, USA, 17-22. DOI=10.1145/1414558.1414564 http://doi.acm.org/10.1145/1414558.1414564
20. Gregory W. Hislop, Heidi J. C. Ellis, and Ralph A. Morelli. 2009. Evaluating student experiences in developing software for humanity. In Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education (ITiCSE '09). ACM, New York, NY, USA, 263-267. DOI=10.1145/1562877.1562959 http://doi.acm.org/10.1145/1562877.1562959
21. Chang Liu. 2005. Enriching software engineering courses with service-learning projects and the open-source approach. In 27th International Conference on Software Engineering (ICSE 2005). IEEE. 2005. 15-21. doi: 10.1109/ICSE.2005.1553612
22. Huiling Ding. 2007. Open Source: Platform for virtual service learning and user-initiated research. In Proceedings of Professional Communication Conference, 2007. IPCC 2007. IEEE International. 1,5 (Oct. 2007), 1-3. doi: 10.1109/IPCC.2007.4464080
23. Steve Jacobs. 2010. Building an education ecology on serious game design and development for the One Laptop Per Child and Sugar platforms: A service learning course builds a base for peer mentoring, cooperative education internships and sponsored research, Games Innovations Conference (ICE-GIC), 2010 International IEEE Consumer Electronics Society's. 1, 6 (Dec. 2010), 21-23. doi: 10.1109/ICEGIC.2010.5716882
24. Bonnie K. MacKellar, Mihaela Sabin, and Allen Tucker. 2013. Scaling a framework for client-driven open source software projects: a report from three schools. J. Comput. Sci. Coll. 28, 6 (June 2013), 140-147.
25. Heidi J. C. Ellis, Ralph A. Morelli, Trishan R. de Lanerolle, Jonathan Damon, and Jonathan Raye. 2007. Can Humanitarian Open-Source Software Development Draw New Students to CS? SIGCSE 2007, Ellis, Heidi JC, et al. "Can humanitarian open-source software development draw new students to CS? ACM SIGCSE Bulletin. 39, ACM, 2007.
26. Heidi J. C. Ellis, Stoney Jackson, Gregory W. Hislop, Darci Burdge and Lori Postner. 2014. Learning Within a Professional Environment: Shared Ownership of an HFOSS Project. In Proceedings of the 15th Annual Conference on Information technology education, 95-100. ACM.
27. Becka Morgan and Carlos Jensen. 2014. Lessons Learned from Teaching Open Source Software Development, Open Source Software: Mobile Open Source Technologies, IFIP Advances in Information and Communication Technology, 427, 133-142.
28. Heidi J. C. Ellis, Lori Postner, Gregory W. Hislop, and Stoney Jackson. 2015. Team Project Experiences in Humanitarian Free and Open Source Software (HFOSS). Special Issue of the ACM Transactions on Computing Education on Team Projects in Computing Education, 15, 2, 2015.